Intel® Pentium® 4 Processor Identification and the CPUID Instruction

Revision - 001

July, 2000

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect IL 60056-7641

or call 1-800-879-4683
or visit Intel's website at http://www.intel.com/

CONTENTS

TABLES

REVISION HISTORY

Revision	Revision History	Date
-001	Original Issue.	07/2000

1 Introduction

This document provides additional information about the CPUID instruction specific to the Intel® Pentium® 4 family of processors. For a detailed description of the P6 family CPUID format, please refer to the *Intel Architecture Software Developer's Manual, Volume 2* or Application Note *AP-485 Intel Processor Identification and the CPUID Instruction.* Software should use the CPUID instruction to identify the Pentium® 4 processors and the features supported by the Pentium® 4 processor.

Two sets of CPUID functions are supported in the Pentium® 4 family of processors: **standard functions**, and **extended functions**.

2 Standard CPUID Functions

The standard CPUID functions are those functions supported by previous generations of Intel Architecture 32-bit processors (IA-32). A standard CPUID functions is invoked by loading the EAX register with a value of 0, 1, or 2 prior to executing the CPUID instruction. The standard functions return results in the general-purpose registers EAX, EBX, ECX, and EDX.

The Pentium® 4 processor does not support function 3 the processor serial number feature introduced by the Pentium® III processor.

2.1 Vendor-ID and Largest Standard Function # (Function 0)

On IA-32 processors, when EAX is initialized to a value of 0, the CPUID instruction returns the largest standard function number supported by the processor in register EAX (refer to Table 1), and the Vendor-ID string in EBX, EDX, and ECX. Software should validate the Vendor-ID before executing additional CPUID functions.

Table 1 - Vendor-ID and Largest Standard Function

	31	23	15	7 0
EAX	Largest standard function number supported			
EBX	u (75h)	n (6Eh)	e (65h)	G (47h)
EDX	I (49h)	e (65h)	n (6Eh)	i (69h)
ECX	l (6Ch)	e (65h)	t (74h)	n (6Eh)

2.2 Feature Information (Function 1)

On IA-32 processors, when EAX is initialized to a value of 1, the CPUID instruction returns the *Type, Family, Model,* and *Stepping* value in the EAX register (see Table 2). The *Feature Flags* are returned in the EDX register (see Table 4).

The *Brand Id* was first introduced by the Pentium® III processor, model 8, and is also supported by the Pentium® 4 processor. When function 1 is invoked on the Pentium® 4 processor, the *Brand Id* is returned in EBX[7:0] (see Table 3).

Beginning with the Pentium® 4 processor, the value returned in EBX[31:24] reflects the default APIC ID. This is the value assigned by the hardware when the processor comes out of RESET#. Note that it is possible for software to modify the APIC_ID but the value returned in EBX[31:24] by function 1 of the CPUID instruction remains unchanged, always indicating the default APIC_ID.

Beginning with the Pentium® 4 processor, the value returned in EBX[15:8] reflects the number of 8-byte chunks flushed by the CLFLUSH instruction. For the Pentium® 4 processor this field returns a value of 8 (64-bytes).

Table 2 - CPUID Feature Information

Register Bits	Description
EAX [31:14]	Reserved
EAX [13:12]	Processor Type = 00b
EAX [11:8]	Family = 1111b
EAX [7:4]	Model = xxxxb, Initially 0000b
EAX [3:0]	Stepping = xxxxb
EBX[31:24]	Default APIC ID
EBX[23:16]	Logical processor count
EBX[15:8]	CLFLUSH chunk count
EBX[7:0]	Brand ID
ECX[31:0]	Reserved
EDX[31:0]	Feature Flags

Table 3 - Pentium® 4 Family Brands

EBX [7:0]	Intel Brand
00001000b	Intel® Pentium® 4 processor
00001001b	To Be Announced
00001010b	To Be Announced
00001011b	To Be Announced
00001100b	To Be Announced
00001101b	To Be Announced
00001110b	To Be Announced
00001111b	To Be Announced

Table 4 - Feature Flag Definitions

Bit	Descriptions
31:30	**Reserved.**
29	**Automatic Clock Control.** A value of 1 indicates the Thermal Monitor automatic thermal control circuit (TCC) is supported by this processor.
28	**Reserved.**
27	**Self-Snoop.** A value of 1 indicates this processor contains hardware to maintain a consistent memory image in the presence of aliased memory types.
26	**SSE2.** A value of 1 indicates the presence of the Streaming SIMD Extensions - 2 (SSE2). A value of 0 indicates the extensions are not present in the processor.
25	**SSE.** A value of 1 indicates the presence of the Streaming SIMD Extensions (SSE). A value of 0 indicates the extensions are not present in the processor.

24	**Fast Save and Restore.** A value of 1 indicates that the processor supports the FXSAVE and FXSTOR instructions for fast save and restore of floating point context. A value of 0 indicates the instructions are not supported in the processor.
23	**MMX™ Technology.** Indicates whether the processor supports the MMX™ technology instruction set and architecture. A value of 1 indicates the presence of the MMX feature. A value of 0 indicates the feature is not present in the processor.
22	**ACPI Support.** A value of 1 Indicates the processor supports clock controlling by software. This bit indicates the processor supports ACPI registers in MSR space and the associated Temperature LVT entry and interrupt polarity MSR.
21	**Debug Trace Store.** Indicates that the processor has the ability to write a history of the branch to and from addresses into a memory buffer. A value of 1 indicates the presence of the Debug Trace Store feature. A value of 0 indicates the feature is not present in the processor.
20	**Reserved.**
19	**CLFLUSH Instruction.** A value of 1 indicates that the processor supports the CLFLUSH instruction. When this bit is 1, the value returned in EBX[15:8] indicates the number of chunks flushed from the cache.
18	**Processor Serial Number.** A value of 1 indicates the presence of the Processor Serial Number feature, and that the feature is enabled. A value of 0 indicates the feature is not present in the processor or is currently disabled (refer to the register definition for MSR_MISC_CTL for details on how to disable this feature). **The Pentium® 4 processor family does not support this feature.**
17	**36-bit Page Size Extension (PSE-36).** A value of 1 indicates the presence of the PSE feature. A value of 0 indicates the feature is not present in the processor. The 36-bit page size extension adds four bits to the 4Mbyte-page descriptor. Indicates whether the processor supports 4-Mbyte pages that are capable of addressing physical memory beyond 4GB. This feature indicates that bits 13-16 of the page directory entry encode the upper four bits of the physical address of the 4-Mbyte page.
16	**Page Attribute Table (PAT).** A value of 1 indicates the presence of the PAT feature. A value of 0 indicates the feature is not present in the processor. Indicates whether the processor supports the Page Attribute Table. This feature augments the Memory Type Range Registers (MTRRs), allowing an operating system to specify attributes of memory on a 4k granularity through a linear address.
15	**Conditional Move Instruction Supported.** The processor supports CMOVcc, and if the FPU feature flag (bit 0) is also set, supports the FCMOVCC and FCOMI instructions.
14	**Machine-Check Architecture.** A value of 1 indicates the Machine Check Architecture is supported, specifically the MCG_CAP register.
13	**Page Global Enable.** A value of 1 indicates the global bit in the PDEs and PTEs and the CR4.PGE enable bit are supported.
12	**Memory Type Range Registers.** A value of 1 indicates the Processor supports the Memory Type Range Registers specifically the MTRR_CAP register.
11	**Fast System Call (SEP).** A value of 1 indicates the presence of the SEP feature. A value of 0 indicates the feature is not present in the processor. Indicates whether the processor supports the Fast System Call instructions, SYSENTER and SYSEXIT.
10	**Reserved.**
9	**On-chip APIC Hardware Supported.** The processor contains a local APIC.
8	**CMPXCHG8 Instruction Supported.** The compare and exchange 8 bytes instruction is supported.
7	**Machine-Check Exception.** Machine Check Exception, INT18, and the CR4.MCE enable bit are supported.
6	**Physical Address Extension.** Physical addresses greater than 32 bits are supported.

5	**Model Specific Registers.** Model Specific Registers are implemented with the RDMSR, WRMSR instructions.
4	**Time Stamp Counter.** The RDTSC instruction is supported including the CR4.TSD bit for access/privilege control.
3	**Page Size Extension.** The processor supports 4-Mbyte pages.
2	**Debugging Extension.** The processor supports I/O breakpoints, including the CR4.DE bit for enabling debug extensions and optional trapping of access to the DR4 and DR5 registers.
1	**Virtual Mode Extension.** The processor supports extensions to virtual-8086 mode.
0	**Floating-point Unit On-chip.** The processor contains an FPU that supports the Intel387 floating-point instruction set.

2.3 Cache Descriptors (Function 2)

When EAX is initialized to a value of 2, the CPUID instruction returns cache and TLB descriptor parameters in the EAX, EBX, ECX and EDX registers.

Table 5 lists the current definitions for the various cache descriptors returned by the Pentium® 4 family of processors.

Table 5 - Pentium® 4 family processors Cache and TLB Descriptors

Cache or TLB Descriptor Description	Descriptor value (Hex)
512K L3 Cache, 4-way, sectored, 64 byte line size	22h
1MB L3 Cache, 8-way, sectored, 64 byte line size	23h
2MB L3 Cache, 8-way, sectored, 64 byte line size	25h
4MB L3 Cache, 8-way, sectored, 64 byte line size	29h
No L3 Cache	40h
64 entries for 4KB pages & 2MB/4MB pages (ITLB)	50h
128 entries for 4KB pages & 2MB/4MB pages (ITLB)	51h
256 entries for 4KB pages & 2MB/4MB pages (ITLB)	52h
64 entries for 4KB pages & 4MB pages (DTLB)	5Bh
128 entries for 4KB pages & 4MB pages (DTLB)	5Ch
256 entries for 4KB pages & 4MB pages (DTLB)	5Dh
8KB L1 Data Cache, 4-way set associative	66h
16KB L1 Data Cache, 4-way set associative	67h
32KB L1 Data Cache, 4-way set associative	68h
12K Trace Cache, 4-way set associative	70h
16K Trace Cache, 4-way set associative	71h
32K Trace Cache, 4-way set associative	72h
128KB L2 Cache, 8-way set associative, sectored, 64 byte line size	79h

256KB L2 Cache, 8-way set associative, sectored, 64 byte line size	7Ah
512KB L2 Cache, 8-way set associative, sectored, 64 byte line size	7Bh
1MB L2 Cache, 8-way set associative, sectored, 64 byte line size	7Ch

3 Extended CPUID Functions

The Pentium® 4 family of processors implement extended CPUID functions. To test for support for extended CPUID functions software should executes the CPUID instruction with EAX = 80000000h. If the CPUID instruction returns a value greater than 80000000h then the extended functions are supported.

The following sections describe what the Pentium® 4 processor returns when the allowable extended function numbers are entered into EAX.

3.1 Largest Extended Function # (Function 80000000h)

On the Pentium® 4 family of processors, when EAX is initialized to a value of 80000000h, the CPUID instruction returns the largest extended function number supported by the processor in register EAX (refer to Table 6).

Table 6 - Largest Extended Function

	31 23 15 7 0
EAX	Largest extended function number supported
EBX	Reserved
EDX	Reserved
ECX	Reserved

3.2 Extended Processor Signature and Extended Feature Bits (Function 80000001h)

On the Pentium® 4 family of processors, when EAX is initialized to a value of 80000001h, the CPUID instruction returns 0 in the registers EAX, EBX, ECX and EDX.

3.3 Processor Name / Brand String (Function 80000002h, 80000003h, 80000004h)

Functions 80000002h, 80000003h, and 80000004h each return up to 16 ASCII bytes of the processor name in the registers EAX, EBX, ECX, and EDX. The processor name is constructed by concatenating each 16-byte ASCII string returned by the three functions. The processor name may be right justified and have leading space characters. It is returned in little endian format and NULL terminated. The maximum length of the processor name is 48 bytes. The 48 bytes that comprise the processor name include the *Brand String*, intended speed in ASCII format and the NULL terminator character.

3.3.1 BUILDING THE PROCESSOR NAME

Software must reserve enough space in a byte array to concatenate the three 16 byte ASCII strings that comprise the processor name. Software must execute each function in sequence. After sequentially executing each CPUID processor name function, software must concatenate EAX, EBX, ECX, EDX to create the resulting processor name string.

```
Processor_Name     db  48 dup(0)

        MOV     EAX, 80000000h
        CPUID
        CMP     EAX, 80000000h     ; Check if extended functions are supported
        JB      Not_Supported
        MOV     EAX, 80000002h
        MOV     DI, OFFSET Processor_Name
        CPUID                      ; Get first 16 bytes of the processor name
```

```
        CALL    Save_String
        MOV     EAX, 80000003h
        CPUID                           ; Get second 16 bytes of the processor name
        CALL    Save_String
        MOV     EAX, 80000004h
        CPUID                           ; Get last 16 bytes of the processor name
        CALL    Save_String

Not_Supported:
        RET

Save_String
        MOV     Dword Ptr [DI], EAX
        MOV     Dword Ptr [DI+4], EBX
        MOV     Dword Ptr [DI+8], ECX
        MOV     Dword Ptr [DI+12], EDX
        ADD     DI, 16
        RET
```

3.3.2 DISPLAYING THE PROCESSOR NAME

The processor name may be a right justified string padded with leading space characters. When displaying the processor name string the display software must skip the leading space characters and discontinue printing characters when the NULL character is encountered.

```
        CLD
        MOV     SI, OFFSET Processor_Name     ; Point SI to the name string

Spaces:
        LODSB
        CMP     AL, ' '                       ; Skip any leading Space chars
        JE      Spaces
        CMP     AL, 0                         ; Exit if NULL byte encountered
        JE      Done

Display_Char:
        CALL    Display_Character             ; Put a char on the output device
        LODSB
        CMP     AL, 0                         ; Exit if NULL byte encountered
        JNE     Display_Char

Done:
```